2

Technical Report
930

An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model

D.B. Paul

18 July 1991

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

Prepared for the Defense Advanced Research Projects Agency
under Air Force Contract F19628-90-C-0002.

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. The work was sponsored by the Tactical Technology Office of the Defense Advanced Research Projects Agency under Air Force Contract F19628-90-C-0002 (DARPA Order Number 3391).

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

AN EFFICIENT A* STACK DECODER ALGORITHM FOR CONTINUOUS SPEECH RECOGNITION WITH A STOCHASTIC LANGUAGE MODEL

D.B. PAUL
Group 24

TECHNICAL REPORT 930

18 JULY 1991

LEXINGTON MASSACHUSETTS

ABSTRACT

The stack decoder is an attractive algorithm for controlling the acoustic and language model matching in a continuous speech recognizer. It implements a best-first tree search to find the best match to both the language model and the observed speech. A previous paper described a near-optimal admissible Viterbi A* search algorithm for use with non-cross-word acoustic models and no-grammar language models [1]. This report extends this algorithm to include unigram language models and describes a modified version of the algorithm which includes the full (forward) decoder, cross-word acoustic models and longer-span language models. The resultant algorithm is not admissible, but has been demonstrated to be very efficient.

TABLE OF CONTENTS

1. INTRODUCTION

Speech recognition may be treated as a tree network search problem. As one proceeds from the root toward the leaves, the branches leaving each junction represent the set of words that may be appended to the current partial sentence. Each of the branches leaving a junction has a probability and each word has a likelihood of being produced by the observed acoustic data. The recognition problem is to identify the most likely path (word sequence, W^*) from the root (beginning of the sentence) to a leaf (end of the sentence) taking into account the junction probabilities (the stochastic language model) $p(W)$ and the acoustic match (including time alignment) $p(O|W)$ given that path [2]

$$W^* = \underset{\{W\}}{\operatorname{argmax}}\, p(O|W)p(W) \tag{1}$$

where O is the acoustic observation sequence and W is a word sequence. Similarly, for no-grammar language model recognition, the problem is to identify the most likely word sequence given only the acoustic data

$$W^* = \underset{\{W\}}{\operatorname{argmax}}\, p(O|W). \tag{2}$$

By Bayes rule, likelihoods can be substituted for the probabilities in either of the above equations without changing the recognized sentence W^*.

This report is concerned with the network search problem and therefore correct recognition is defined as outputting the most likely sentence W^* given the language model, the acoustic models, and the observed acoustic data. If the most likely sentence is not the one spoken, it is a modeling error—not a search error. This paper will assume for simplicity that an isolated sentence is the object to be recognized. (The algorithm extends trivially to recognize continuous input.)

This report will assume a stochastic acoustic model, such as a hidden Markov model (HMM) [2,3,4], and a stochastic language model [5]. An accept-reject language model will also work—its output log-likelihood is either zero or minus infinity.

This report will initially describe the basic stack decoder and the A* scoring criterion. It will then describe a near-optimal admissible search algorithm for using the (Viterbi) stack decoder to perform continuous speech recognition (CSR) with a no-grammar/unigram language model. Finally, it will show how to modify this algorithm to produce an efficient stack decoder that includes the full (forward) decoder, cross-word acoustic models, and longer-span language models.

2. THE BASIC STACK DECODER

The stack decoder [6], as used in speech, is an implementation of a best-first tree search. The basic operation of a sentence decoder is as follows [2,7]:

1. Initialize the stack with a null theory.
2. Pop the best (highest scoring) theory off the stack.
3. if(end-of-sentence) output the sentence and terminate.
4. Perform acoustic and language-model fast matches to obtain a short list of candidate word extensions of the theory.
5. For each word on the candidate list:
    - (a) Perform acoustic and language-model detailed matches to compute the new theory output log-likelihood.
        - i. if(not end-of-sentence) insert into the stack.
        - ii. if(end-of-sentence) insert into the stack with end-of-sentence flag = TRUE.
6. Go to 2.

The fast matches [7,8] are computationally cheap methods for reducing the number of word extensions that must be checked by the more accurate, but computationally expensive detailed matches.[1] (The fast matches may also be considered a predictive component for the detailed matches.) Top-N (N-best) mode is achieved by delaying termination until N sentences have been output.

The stack itself is just a sorted list that supports the following operations: pop the best entry and insert new entries according to their scores. The following items must be contained in the ith stack entry:

1. A stack score: $StSc_i$
2. A reference time: t_ref_i
3. A word history (path or theory identification)
4. An output log-likelihood distribution: $L_i(t)$
5. An end-of-sentence flag

[1]The following discussion concerns the basic stack decoder and therefore it will be assumed that the correct word will always be on the fast match list. This can be guaranteed by the scheme outlined in Bahl et al [7]. All of the theoretical results contained in this report assume no fast matches.

The stack score and the reference time are used to sort the stack. Since the time of exiting the last word of the theory cannot be uniquely determined without knowing the next word, the output log-likelihood as a function of time must be contained in each entry. This distribution is the input to the next word model. The end-of-sentence flag identifies the theories that are candidates to end the sentence.

3. THE A* STACK CRITERION

A key issue in the stack decoder is deciding which theory should be popped from the stack to be extended. This is decided by the stack score and the reference time. (All scores used here are log-likelihoods or log-probabilities.) If one uses the raw log-probabilities as the stack score, a *uniform search* [9] will result. This search will result in a prohibitive amount of computation and a very large stack because the log-probabilities decrease rapidly with path length and thus short paths will be "carried along" by the better paths. A better scoring criterion is the *A* criterion* [9]. The (near-optimal) A* criterion used here is the difference between the actual log-likelihood of reaching a point in time on a path and an upper bound on the log-likelihood of any path reaching that point in time:

$$\Lambda_i(t) = L_i(t) - \text{ub}L(t) \tag{3}$$

where $\Lambda_i(t)$ is the A* scoring function, $L_i(t)$ is the output log-likelihood, t denotes time, i denotes the path (tree branch or left sentence fragment) and ub$L(t)$ is an upper bound on $L_i(t)$. (This criterion is similar to the exact A* criterion stated in Nilsson [9] except that, unlike the exact A* criterion, it can be computed at negligible cost without using data from the future. See Appendix A for a derivation.) In order to sort the stack entries, it is necessary to reduce the $\Lambda_i(t)$ to a single number (the stack score):

$$StSc_i = \max_t \Lambda_i(t). \tag{4}$$

It is also convenient at this point to define the *minimum* time that satisfies Equation (4):

$$t\_min_i = \underset{t}{\text{argmin}}\,(StSc_i = \Lambda_i(t)). \tag{5}$$

Ideally, ub$L(t)$ would be the least upper bound on $L_i(t)$: lub$L(t)$. In general, the closer ub$L(t)$ is to lub$L(t)$, the less computation. If ub$L(t)$ becomes less than lub$L(t)$, longer paths will be favored excessively and the first output sentence may not have the highest log-likelihood, i.e., a search error may occur. (Note that ub$L(t)$ is constant for any t and therefore does not affect the relative scores of the paths at any fixed time—it only affects the comparison of paths of differing lengths and the resultant order of path expansion.)

The stack search will be admissible [9] if the following conditions are met:

$$\text{ub}L(t) \geq \text{lub}L(t) \tag{6}$$

and

$$\text{ub}L(t_2) - \text{lub}L(t_2) \geq \text{ub}L(t_1) - \text{lub}L(t_1) \qquad t_2 \geq t_1 \tag{7}$$

and it will be near-optimal [9] ir the sense that a near-minimum number of theories will need to be expanded for an admissible search if

$$\text{ub}L(t) = \text{lub}L(t). \tag{8}$$

The additional condition stated in Equation (7) over those found in Nilsson [9] is sufficient (but not necessary) because the word acoustic models can "jump" over parts of the bound and therefore a locally poor bound can block the correct theory while passing those that are able to jump over the poor region.

A basic problem is obtaining a good estimate of ub$L(t)$ in a time-asynchronous decoder. (Note that lub$L_{state}(t)$ over the states is easily computed in a time-synchronous decoder and that $\Lambda_{state}(t)$ is the value compared to the pruning threshold in a beam search [10].) One simple estimate of lub$L(t)$ is

$$\hat{\text{lub}}L(t) = -\alpha t. \tag{9}$$

where α is some constant greater than zero. This approach attempts to cancel out the average log-likelihood per time step. If α is too large, it will underestimate the bound and risk recognition errors. If α is small enough. the search will be admissible but will require an excessive amount of computation. (In fact, $\alpha = 0$ is the uniform search mentioned above.) Unfortunately no single value of α is optimum for all sentences or all parts of a single sentence. Thus a conservative value must be chosen and the computation will be excessive.

This estimate of the bound is not useful for controlling the stack decoder, but it is adequate to estimate the most-probable theory exit time. Given an appropriate value for α, $\Lambda_i(t)$ will exhibit a peak whose location is an estimate of the most-probable exit time of the theory. (This stack decoder only implements the forward decoder—finding the exact most-probable exit time requires information from the decode of some amount of the remainder of the sentence.) Therefore the estimated exit time is

$$\hat{t}\_exit_i = \underset{t}{\text{argmax}}\ L_i(t) - \alpha t. \tag{10}$$

4. THE NEAR-OPTIMAL A* STACK DECODER FOR RECOGNITION WITH A NO-GRAMMAR OR UNIGRAM LANGUAGE MODEL

It is not possible to compute the exact least upper bound on the theory likelihoods without first performing the recognition. It is, however, possible to compute the least-upper-bound-so-far (lubsf) on the likelihoods that have already been computed, which requires negligible computation and is to be sufficient to perform the near-optimal A* search. This creates two problems:

1. Since $\text{lub}L(t) = \text{lubsf}L(t)$ can change as the theories are evaluated, the stack order can also change.
2. A degeneracy in determining the best path by $StSc$ alone can occur because $\text{lubsf}L(t)$ can equal $L_i(t)$ for more than one i (path) at different times.

Problem 1 is easily cured by reevaluating the stack score $StSc$ every time $\text{lubsf}L(t)$ is updated and reorganizing the stack. This is easily accomplished if the stack is stored as a heap [11].

Problem 2 occurs because different theories may dominate different parts of the current upper bound. Thus all of these theories will have a score of zero. If the longest theory is extended, its descendents will in turn dominate the longest part of the bound and will therefore be extended. This will, of course, result in search errors because the shorter theories will never have a chance to be extended. The cure is to extend the shortest theory (minimum t_min) that has a stack score equal to the best. If $t\_ref_i = t\_min_i$, this can be accomplished by performing a major sort on the stack score $StSc$ and a minor sort on the reference time t_ref.

This guarantees that $\text{lubsf}L(t) = \text{lub}L(t)$ for $t \leq t\_ref_p$ (*where p denotes the theory that is* about to be popped) and therefore the relevant part of the least-upper-bound has been computed by the time that it is needed. Since the bound, at the time that it is need, is the least-upper-bound, the search is admissible and near-optimal. Furthermore, when the first sentence is output, the least-upper-bound-so-far will be the exact least-upper-bound. (This assumes no fast match was used. An aggressive fast match can cause some portions of the lubsf to underestimate the exact lub but this will not destroy the admissibility of the search as long as the correct word is on the fast match list.)

A stack pruning threshold can be used to limit the stack size [1]. Any theory whose $StSc$ falls below the threshold can be deleted from the stack. This can be applied on stack insertions and any time the stack is reorganized. (Any update to the lubsf will only cause the $StSc$ to be reduced, so any theory that is pruned will not be accepted by the pruning threshold at a later portion of the search.) This stack pruning threshold has little effect on the computational requirements and can therefore be set very conservatively to essentially eliminate any chance that the correct theory will be pruned.

In a time-synchronous (TS) no-grammar/unigram language model Viterbi decoder, all word output likelihoods are compared and only the maximum is passed on as input to the word models. Thus by comparison, only theories that dominate the lubsf need be retained on the stack and the stack pruning threshold can be set to zero for top-1 recognition. Since all stack scores, $StSc$, of all theories popped from the stack will be zero until the first sentence is output, all theories popped from the stack will be in reference time t_min order. (Of course, the stack pruning threshold must be nonzero if a top-N list of sentences is desired.) For top-N recognition, this algorithm adaptively raises the effective computational pruning threshold (which equals the current best $StSc$) by the minimum required to produce N output sentences, subject to the limit placed by the stack pruning threshold.

A time-synchronous Viterbi decoder must sometimes make an arbitrary decision when two output likelihoods are equal, as can be caused by homonyms with identical acoustic models. The top-1 stack decoder will maintain both theories and, if all theories with likelihood equal to that of the best theory are output, all homonym variations will be output. (This does not require a nonzero stack pruning threshold because both theories will have the same likelihood.)

As is shown in Appendix A, this algorithm is near-optimal and admissible only for a Viterbi decode using non-cross-word accustic models and a no-grammar or unigram language model. In a full (forward) decode, the output likelihood of one theory—which may later have a higher likelihood due to summing of the probabilities of paths with different time alignments—may be "shadowed" by the output likelihood of a second theory. The first theory would therefore not be expanded. In general, the recognition accuracy of the Viterbi decoder is very similar to the accuracy of the full decoder. (Small differences have been observed in some experiments.)

The long-span language model flaw in this algorithm can similarly cause shadowing of the correct theory [12]. A language model can improve the relative likelihood of a theory such that a currently shadowed theory can dominate the bound after expansion. The above algorithm will not allow the shadowed theory to be expanded until all better (higher stack score) theories have been expanded. Such will occur if the system is run in top-N mode for a large enough N. Similarly an end-of-sentence penalty can force a shadowed theory to be expanded. Unfortunately both of these methods greatly increase the computation.

5. AN EFFICIENT STACK DECODER ALGORITHM FOR USE WITH A MULTIPLE-WORD SPAN LANGUAGE MODEL

An efficient stack decoder algorithm that can be used with cross-word acoustic models, the full (forward) decoder, and longer-span (≥ 2) language models can be produced by two simple changes:

1. Change the stack ordering to be a major sort on the reference time t_ref (favoring the lesser times) and a minor sort on the stack score $StSc$.
2. Use a non-zero stack pruning threshold.

The reference time t_ref may also be changed from the minimum time that satisfies Equation (4) used in the no-grammar/unigram language-model version to $\hat{t}\_exit$ as defined in Equation (10). (Either will work and both required similar amounts of computation in tests.) This algorithm appears to be a simplification of one developed at IBM [13]. This algorithm is not admissible because the correct theory can be pruned from the stack. The stack-pruning threshold now controls the trade-off between the amount of computation and the probability of pruning the correct theory by controlling the likelihood "depth" that will be searched. Unlike the previous algorithm, an (unpruned) theory will not be shadowed because it will be extended when its reference time is reached. This algorithm is quasi-time-synchronous because it, in effect, moves a time bound forward and whenever this time bound becomes equal to the reference time of a theory, the theory is expanded. (Descendants of the theory will generally have reference times farther forward in time.)

Note that the stack pruning threshold can also be set to zero for no-grammar/unigram language model top-1 recognition with this algorithm. With a zero stack pruning threshold and $t\_ref_i = t\_min_i$, it becomes equivalent to the near-optimal, admissible no-grammar/unigram language model algorithm described above for top-1 recognition. (While this algorithm can also perform top-N recognition with or without a language model, it cannot be made equivalent to the no-grammar/unigram language model version for top-N. Its pruning threshold is fixed and it will only output theories whose relative likelihoods do not fall below the threshold.)

Given that this stack decoder algorithm has become quasi-time-synchronous, what are its advantages over a time synchronous system?

1. It efficiently combines a long-span language model with the acoustic recognition into the search.
2. It is an effective control strategy for simultaneous interpretation of large language and acoustic models.
3. It has a tighter pruning threshold than a TS system because the threshold is applied only to word end likelihoods after the language model has been applied rather than to all states (which includes word internal states as well as states immediately before and after application of the language model).

4. It may require fewer applications of the fast match since the fast match need only be applied once per popped theory rather than once per time step. (This can be guaranteed by an appropriately chosen fast match algorithm.)
5. It produces a top-N sentence output with only negligible modification to the algorithm.
6. It can perform a Viterbi decode or an exact full decode without difficulty. The likelihood "depth" of the search combined with the time-based major (stack) sort allows pragmatically admissible operation of the full decode.

6. DISCUSSION AND CONCLUSIONS

The stack-search algorithms discussed in this report have been implemented in a prototype that uses real speech input, but does not yet have all of the features of the Lincoln TS CSR [14,15]. (The primary missing features are cross-word phonetic modeling and tied-mixtures.) The prototype runs faster than does the TS system on the corresponding recognition task, frequently by a significant factor. (In fairness, the TS system does not include a fast match.) Current experience using the DARPA Resource Management Database [16] shows the required number of stack pops and the stack size to be surprisingly small. In addition, the prototype includes a proposed CSR - NL interface [17] and has been run with unigram, word-pair, bigram, and trigram language models accessed through the interface without difficulty. (It has also been run using a no-grammar language model, which, of course, does not require the interface.)

Methods for joining the acoustic matching of separate theories and caching of acoustic computations to reduce the acoustic match computation were described in Paul [1]. These algorithms were tested in a stack-decoder simulator (real stack decoder with simulated input data). These accelerators have not been tested in the prototype stack decoder, but there is no reason in principle why they could not be used.

A* search using the scoring function described by Nilsson [9] [Equation (A.1)] requires computing the likelihood of the future data ($h^*(t)$ in Equation (A.4). The optimal A* decoder requires exact evaluation of $h^*(t)$. which requires solving the top-1 recognition problem by some other means, such as a reverse direction TS decoder [18], before the A* search can begin. The alternative described here substitutes a near-optimal scoring function that is derived from the A* search and requires negligible additional computation over that required by the search itself. Since, as noted above, the Lincoln top-1 TS decoder takes more CPU time than does the near-optimal stack decoder, the near-optimal stack decoder algorithm appears to be the most efficient of the three approaches for top-1 recognition. In addition, the inadmissible version of the near-optimal stack decoder can very easily integrate long-span language models into the search. However, if top-N recognition is the goal, the optimal A* search may be preferred because, once the price is paid for computing $h^*(t)$, the A* search can find the additional N-1 sentences very efficiently for no-grammar/unigram language models [18]. A longer span language model would require computing $h_i^*(t)$ [Equation (A.1)] rather $h^*(t)$ [Equation (A.4)], which would increase the cost of computing the additional sentences.

Recently several other algorithms have been proposed for top-N recognition using A* search [19,18,20] that use the Nilsson formulation of the scoring function. All of these approaches use a reverse direction TS decoder to compute $h^*(t)$. (There are also some proposed non-A* methods for recognizing the top-N sentences [21,22,23]. In general, the bidirectional approaches appear to be more efficient than the unidirectional approaches.) These A* (and bidirectional) methods must wait for the end of data (or a pseudo-end-of-data) to begin the A* (or the reverse direction) pass. In contrast, because they do not need data beyond that necessary to extend the current theory (this includes data up to *t_ref* required to choose the current theory), the two stack decoder formulations proposed here can proceed totally left-to-right as the input data becomes available from the front end. The multiple-word-span language-model version of the stack search will output all top-N theories with minimal delay following the end-of-data because all theories are pursued in quasi-parallel or, in top-1 mode, it can output the partial sentence as soon as all unpruned theories have a common partial history (initial word sequence). (A similar technique for continuous output after a short delay from continuous input exists for TS decoders [24]).

One of the motivations for some of these other A* (and top-N) algorithms is as a method for using weaker and cheaper initial acoustic and language models to produce a top-N sentence list for later refinement by more detailed and expensive acoustic and/or language models, which now need only consider a few theories. In contrast the algorithm proposed here integrates both the detailed acoustic and language models directly in the stack search and therefore need only produce a top-1 output. It attempts to minimize the computation by applying all available information to constrain the search. (The stack decoder as described here can, of course, also be used with weak and cheap acoustic and/or language models to produce a top-N list for later processing.) The ultimate choice between the two methods may be determined by the number of sentences required by the top-N approaches and the relative computational costs of the various modules in each system. The architectural simplicity of each system may also have some bearing.

The stack decoder has long shown promise for integrating long-span language models and acoustic models into a single effective search which applies information from both sources into controlling the search. It has not been used at many sites, primarily due to the difficulty of making the search efficient. The algorithms described above will hopefully remove this barrier.

APPENDIX A
DERIVATION OF THE A* CRITERION USED IN EQUATION (3)

Nilsson [9] states the A* criterion (slightly rewritten to match the speech recognition problem) as

$$f_i(t) = g_i(t) + h_i^*(t) \tag{A.1}$$

where $f_i(t)$ is the log-likelihood of a sentence with the partial theory i ending at time t, $g_i(t)$ is the log-likelihood of partial theory i, and $h_i^*(t)$ is the log-likelihood of the best extension of theory i from time t to the end of the data. (Nilsson uses costs which are interpreted here as negative log-likelihoods. All descriptions here will use sign conventions appropriate for log-likelihoods to be consistent with the rest of the paper.) The theory $\underset{i}{\operatorname{argmax}} (\max_t f_i(t))$ is chosen as the next to be popped from the stack and expanded.

Equation (A.1) requires that the computation of the total likelihood of a sentence must be separable into a beginning part and an end part separated by a single time, which disallows this derivation for the full (forward) decoder because the full decoder does not have a unique transition time between two words. Thus, the derivation is limited to a decoder which is Viterbi between words. To allow the $h^*(t)$ terms to cancel in Equation (A.5), define $h_i^*(t)$ to be independent of the theory

$$h^*(t) = h_i^*(t). \tag{A.2}$$

Therefore Equation (A.1) can be rewritten as

$$f_i(t) = g_i(t) + h^*(t). \tag{A.3}$$

This requirement limits the derivation to non-cross-word acoustic model and no-grammar or unigram language model recognition tasks.

Define

$$f^*(t) = g^*(t) + h^*(t) \tag{A.4}$$

for the best theory with a word transition at time t. The function $f^*(t)$ is slowly varying with global maxima at the word transition points of the correct theory, at which points it equals the likelihood of the correct theory. Specifically, it is maximum at $t = 0$ and $t = T$. (T is the end of data.) Since

$g_i(t)$ is an exact value (rather than a bound or estimate) for a tree search, $g^*(t) = \text{lub}g_i(t)$ and since $h^*(t)$ is not a function of i, $f^*(t) = \text{lub}f_i(t)$.

Subtract Equation (A.4) from Equation (A.3) and define $\hat{f}_i(t)$

$$\hat{f}_i(t) = f_i(t) - f^*(t) = g_i(t) - g^*(t). \tag{A.5}$$

This is just Equation 3 in a different notation: $g_i(t) = L_i(t)$ and $g^*(t) = \text{ub}L(t)$ (specifically $\text{lub}L(t)$) and therefore $\hat{f}_i(t) = \Lambda_i(t)$. Thus, if $f^*(t)$ were a constant, $\hat{f}_i(t)$ would just be an offset from $f_i(t)$ and the search would be optimum because $\underset{i}{\text{argmax}}\ (\underset{t}{\max}\ \hat{f}_i(t))$ would always be equal to $\underset{i}{\text{argmax}}\ (\underset{t}{\max}\ f_i(t))$. As noted earlier, $f^*(t)$ has maxima at word transition times of the correct theory. Thus $\hat{f}_i(t)$ is zero at word transition times on the correct theory and ≤ 0 for all other i and t. Thus the search is admissible because it can never block the correct theory by giving a better score to an incorrect theory, but sub-optimal because it can cause incorrect theories to be popped from the stack and be evaluated. Since the evaluation function "error" $f^*(t) - f^*(0)$ is slowly varying, the search is near-optimal.

Since the stack decoder treats each theory and all points on the likelihood distribution $L_i(t)$ as a unit, each theory is evaluated at its optimum point: the $\underset{t}{\max}\ \Lambda_i(t)$ as defined in Equation 4, to give it its "best" chance and then, for efficiency, the likelihood of all points on the distribution $L_i(t)$ are extended in one operation.

The fact that all $StSc_i$ are zero until the first sentence is output and the tie is broken by choosing the theory with the minimum reference time t_min ensures that all candidate theories that might alter $\text{lubsf}L_i(t \leq t\_min_p)$ have already been computed. Thus the $\text{lubsf}L(t) = \text{lub}L(t)$ for $t \leq t\_min_p$.

This derivation shows the stack criterion $\max StSc_i$ with a minimum t_min_i tie-breaker to be adequate to perform a near-optimal admissible A*-search Viterbi-recognition with a no-grammar or unigram language-model using a stack decoder algorithm.

REFERENCES

1. D. B. Paul, "Algorithms for an optimal A* search and linearizing the search in the stack decoder," Int. Conf. Acoust., Speech, and Signal Process. 1991, Toronto, Canada, May 14-17 (1991), pp. 693-696.
   also
   D. B. Paul, "Algorithms for an optimal A* search and linearizing the search in the stack decoder," *Proc. June 1990 DARPA Speech and Natural Language Workshop*, San Mateo, CA: Morgan Kaufmann Publishers (1990), pp. 200-204.
2. L. R. Bahl, F. Jelinek, and R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-5, 179-190 (1983).
3. D. B. Paul, "Speech recognition using hidden Markov models," *Lincoln Laboratory Journal*, Vol. 3, No. 1, pp. 41-62 (1990).
4. A. B. Poritz, "Hidden Markov models: A guided tour," Int. Conf. Acoust., Speech, and Signal Process. 1988, April 11-14 (1988), pp. 7-13.
5. F. Jelinek, "Self-organized language modeling for speech recognition," *Readings in Speech Recognition*, A. Weibel and K. F. Lee, eds., San Mateo, CA: Morgan Kaufmann Publishers (1990), pp. 450-506.
6. F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. Develop.*, Vol. 13, 675-685 (1969).
7. L. Bahl, P. S. Gopalakrishnam, D. Kanevsky, D. Nahamoo, "Matrix fast match: A fast method for identifying a short list of candidate words for decoding," Int. Conf. Acoust., Speech, and Signal Process. 1989, Glasgow, Scotland, May 23-26 (1989), pp. 345-348.
8. L. S. Gillick and R. Roth, "A rapid match algorithm for continuous speech recognition," in *Proc. June 1990 DARPA Speech and Natural Language Workshop*, San Mateo, CA: Morgan Kaufmann Publishers (1990), pp. 170-172.
9. N. J. Nilsson, *Problem-Solving Methods of Artificial Intelligence*, New York City, NY: McGraw-Hill (1971), pp. 57-65.
10. B. T. Lowerre, "The HARPY speech recognition system," Ph.D. Thesis, Computer Science Department, Carnegie Mellon University (April 1976).
11. D. E. Knuth, *The Art of Computer Programming: Sorting and Searching*, Vol. 3, Menlow Park, CA: Addison-Wesley (1973), pp. 145-147.
12. J. K. Baker, personal communication (June 1990).
13. L. R. Bahl and F. Jelinek, "Apparatus and method for determining a likely word sequence from labels generated by an acoustic processor," U.S. Patent No. 4,748,670 (31 May 1988).

REFERENCES (Continued)

14. D. B. Paul, "New results with the Lincoln tied-mixture HMM CSR system," *Proc. Fourth DARPA Speech and Natural Language Workshop*, San Mateo, CA: Morgan Kaufmann Publishers (1991).

15. D. B. Paul, "The Lincoln tied-mixture HMM continuous speech recognizer," Int. Conf. Acoust., Speech, and Signal Process. 1991, Toronto, Canada, May 14-17 (1991), pp. 329-332.

16. P. Price, W. Fisher, J. Bernstein, and D. Pallett, "The DARPA 1000-word resource management database for continuous speech recognition," Int. Conf. Acoust., Speech, and Signal Process. 1988, New York City, NY, April 11-14 (1988), pp. 651-654.

17. D. B. Paul, "A CSR-NL interface specification," *Proc. October 1989 DARPA Speech and Natural Language Workshop*, San Mateo, CA: Morgan Kaufmann Publishers (1989), pp. 203-214.

18. F. K. Soong and E. F. Huang, "A tree-trellis fast search for finding the N best sentence hypotheses in continuous speech recognition," Int. Conf. Acoust., Speech, and Signal Process. 1991, Toronto, Canada, May 14-17 (1991), pp. 705-708.

19. P. Kenny, R. Hollan, V. Gupta, M. Lennig, P. Mermelstein, and D. O'Shaughnessy, "A* - admissible heuristics for rapid lexical access," Int. Conf. Acoust., Speech, and Signal Process. 1991, Toronto, Canada, May 14-17 (1991), pp. 689-692.

20. V. Zue, J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, and S. Seneff, "Integration of speech recognition and natural language processing in the MIT voyager system," Int. Conf. Acoust., Speech, and Signal Process. 1991, Toronto, Canada, May 14-17 (1991), pp. 713-716.

21. S. Austin, R. Schwartz, and P. Placeway "The forward-backward search algorithm," Int. Conf. Acoust., Speech, and Signal Process. 1991, Toronto, Canada May 14-17 (1991), pp. 697-700.

22. R. Schwartz and S. Austin, "A comparison of several approximate algorithms for finding multiple (N-best) sentence hypotheses," Int. Conf. Acoust., Speech, and Signal Process. 1991, Toronto, Canada, May 14-17 (1991), pp. 701-704.

23. V. Steinbiss, "Sentence-hypothesis generation in a continuous speech recognition system," EUROSPEECH 89, Paris, France (1989), pp. 51-54.

24. J. C. Spohrer, P. F. Brown, P. H. Hochschild, and J. K. Baker, "Partial backtrace in continuous speech recognition," *Proc. Int. Conf. on Systems, Man, and Cybernetics* (1980), pp. 36-42.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE 18 July 1991	3. REPORT TYPE AND DATES COVERED Technical Report

4. TITLE AND SUBTITLE An Efficient A* Stack Decoder Algorithm for Continous Speech Recognition with a Stochastic Language Model	5. FUNDING NUMBERS C — F19628-90-0002 PE — 62301E, 61101E PR — 337
6. AUTHOR(S) Douglas B. Paul	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lincoln Laboratory, MIT P.O. Box 73 Lexington, MA 02173-9108	8. PERFORMING ORGANIZATION REPORT NUMBER TR-930
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency, 1400 Wilson Blvd. Arlington, VA 22209	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESD-TR-91-111
11. SUPPLEMENTARY NOTES None	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

The stack decoder is an attractive algorithm for controlling the acoustic and language model matching in a continuous speech recognizer. It implements a best-first tree search to find the best match to both the language model and the observed speech. A previous paper described a near-optimal admissible Viterbi A* search algorithm for use with non-cross-word acoustic models and no-grammar language models [1]. This report extends this algorithm to include unigram language models and describes a modified version of the algorithm which includes the full (forward) decoder, cross-word acoustic models and longer-span language models. The resultant algorithm is not admissible, but has been demonstrated to be very efficient.

14. SUBJECT TERMS stack decoder speech recognition continuous speech recognition	A* search hidden Markov models (HMM) language modeling		15. NUMBER OF PAGES 24 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR